IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Eric Anderson	Examiner: Mark A. Radtke
Serial No.:	10/699,486	Group Art Unit: 2165
Filed:	October 31, 2003	Docket No.: 200207252-1
Title:	Textual Filesystem Interface Method and Apparatus	

APPEAL BRIEF UNDER 37 C.F.R. § 41.37

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is filed in response to the Final Office Action mailed October 30, 2007 and Notice of Appeal filed on January 30, 2008.

AUTHORIZATION TO DEBIT ACCOUNT

It is believed that no extensions of time or fees are required, beyond those that may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, such extensions are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required (including fees for net addition of claims) are hereby authorized to be charged to Hewlett-Packard Development Company's deposit account no. 08-2025.

I. REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 20555 S.H. 249 Houston, TX 77070, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

II. RELATED APPEALS AND INTERFERENCES

There are no known related appeals, judicial proceedings, or interferences known to appellant, the appellant's legal representative, or assignee that will directly affect or be directly affected by or have a bearing on the Appeal Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1 – 2 and 4 – 24 are pending in the application and stand finally rejected. Claim 3 was canceled. The rejection of claims 1 – 2 and 4 – 24 is appealed.

IV. STATUS OF AMENDMENTS

No amendments were made after receipt of the Final Office Action. All amendments have been entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in each of the claims involved in the appeal, referring to the specification by page and line number and to the drawings by reference characters, as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified by a corresponding reference to the specification and drawings where applicable. Note that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element or that these are the sole sources in the specification supporting the claim features.

Claim 1

A method of creating a filesystem with transaction based functionality, comprising (Figure 3 shows a flow chart for creating a transactional filesystem with a textual interface: p. 9, lines 1-3 of paragraph [0024].):

receiving an indicator to initiate a transaction for files stored in one or more portions of the filesystem (Figure 3, block 302: Receive an indicator to make directories under a pseudo-filesystem and start a transaction. See p. 9, lines 3-7 of paragraph [0024].);

duplicating the filesystem within a pseudo-filesystem (Figure 3, block 304: Duplicate the filesystem within the pseudo-filesystem. A copy of the filesystem is created and mounted under the pseudo-file system. See p. 10, lines 1-3 of paragraph [0025].);

creating a control text file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem (Figure 3, block 306: Create control and status text files associated with the transaction to receive and process textual information. A user enters text commands into the control text file. See p. 10, lines 1-4 of paragraph [0027].);

processing the text-based commands written to the control text file (Figure 3, block 308: Process the control and status files associated with the transaction. See p. 11, lines 1-3 of paragraph [0028].); and

operating on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands (Figure 3, block 310: Operate on the pseudo-

filesystem within the transaction and according to the text-based commands in the control file. See p. 11, lines 7-11 in paragraph [0028].).

Claim 12

A method of interfacing with a filesystem comprising (Figure 4 is a flow diagram of operations associated with interfacing to a filesystem: see p. 13, lines 1-2 of paragraph [0034]. Figure 5 is a flow diagram of operations with performing a commit command entered into the control text file: see p. 15, lines 1-3 of paragraph [0040].):

receiving a text-based command in a command file for operating on a pseudo-filesystem corresponding to the filesystem within a transaction (Figure 4, block 402: Receive a command indication in a control text file. A user enters a command in a control text file: see p. 13, lines 3-5 of paragraph [0034].);

determining whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem (Figure 4, block 406: Determine whether data dependencies prevent the command from being executed. See p. 14, lines 1-2 of paragraph [0036].);

performing the text-based command to modify a file in the pseudo-filesystem (Figure 4, block 410: Perform the command to affect files associated with the pseudo-filesystem. See p. 15, lines 1-2 of paragraph [0039].);

updating the filesystem to include modifications performed to the file in the pseudo-filesystem (Figure 5, block 510: Perform the commit for files in the pseudo-filesystem and corresponding file system. See p. 16, lines 1-5 in paragraph [0043].); and

updating a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem (Figure 5, block 512: Enter status in to status file to indicate result of commit. See p. 16, lines 7-9 of paragraph [0043].).

Claim 13

The method of claim 12 further comprising:

creating an entire copy of the filesystem (an entire copy of the filesystem is created and mounted under the pseudo-filesystem: see p. 10, lines 1-3 of paragraph [0025].);

mounting the entire copy of the filesystem under the pseudo-filesystem (an entire copy of the filesystem is created and mounted under the pseudo-filesystem: see p. 10, lines 1-3 of paragraph [0025].).

Claim 19

A computer program product for creating a filesystem with transaction based functionality, tangibly stored on a computer readable medium, comprising instructions operable to cause a programmable processor to (Figure 3 shows a flow chart for creating a transactional filesystem with a textual interface: p. 9, lines 1-3 of paragraph [0024].):

receive an indicator to initiate a transaction for files stored in one or more portions of the filesystem (Figure 3, block 302: Receive an indicator to make directories under a pseudo-filesystem and start a transaction. See p. 9, lines 3-7 of paragraph [0024].);

duplicate the filesystem within a pseudo-filesystem (Figure 3, block 304: Duplicate the filesystem within the pseudo-filesystem. A copy of the filesystem is created and mounted under the pseudo-file system. See p. 10, lines 1-3 of paragraph [0025].);

create a control file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem (Figure 3, block 306: Create control and status text files associated with the transaction to receive and process textual information. A user enters text commands into the control text file. See p. 10, lines 1-4 of paragraph [0027].);

process the text-based commands written to the control file (Figure 3, block 308: Process the control and status files associated with the transaction. See p. 11, lines 1-3 of paragraph [0028].); and

operate on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands (Figure 3, block 310: Operate on the pseudo-filesystem within the transaction and according to the text-based commands in the control file. See p. 11, lines 7-11 in paragraph [0028].).

Claim 20

A computer program product for interfacing with a filesystem, tangibly stored on a computer readable medium, comprising instructions operable to cause a programmable processor to (Figure 4 is a flow diagram of operations associated with interfacing to a filesystem: see p. 13, lines 1-2 of paragraph [0034]. Figure 5 is a flow diagram of operations with performing a commit command entered into the control text file: see p. 15, lines 1-3 of paragraph [0040].):

receive a text-based command in a command file for operating on a pseudo-filesystem that is a copy of the filesystem (Figure 4, block 402: Receive a command indication in a control text file. A user enters a command in a control text file: see p. 13, lines 3-5 of paragraph [0034].);

determine whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem (Figure 4, block 406: Determine whether data dependencies prevent the command from being executed. See p. 14, lines 1-2 of paragraph [0036].);

perform the text-based command to update the pseudo-filesystem (Figure 4, block 410: Perform the command to affect files associated with the pseudo-filesystem. See p. 15, lines 1-2 of paragraph [0039].);

update the filesystem to include modifications made to the pseudo-filesystem according to the text-based command (Figure 5, block 510: Perform the commit for files in the pseudo-filesystem and corresponding file system. See p. 16, lines 1-5 in paragraph [0043].); and

update a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem (Figure 5, block 512: Enter status in to status file to indicate result of commit. See p. 16, lines 7-9 of paragraph [0043].).

Claim 21

An apparatus that creates a filesystem with transaction based functionality comprising (Figure 6 is a block diagram of a system 600 for performing methods of the present invention: see p. 16, lines 1-2 of paragraph [0044].):

a processor (Figure 6, block 606: p. 16, line 5 of paragraph [0044].);

a memory (Figure 6, block 602 shows a memory: p. 16, line 3 of paragraph [0044].) having instructions capable of being executed on the processor that receive an indicator to initiate a transaction for files stored in one or more portions of the filesystem (Figure 3, block 302: Receive an indicator to make directories under a pseudo-filesystem and start a transaction. See p. 9, lines 3-7 of paragraph [0024].), duplicate the filesystem within a pseudo-filesystem (Figure 3, block 304: Duplicate the filesystem within the pseudo-filesystem. A copy of the filesystem is created and mounted under the pseudo-file system. See p. 9, lines 1-3 of paragraph [0025].), create a control file that provides a textual filesystem interface for receiving text-based commands to operate on the pseudo-filesystem (Figure 3, block 306: Create control and status text files associated with the transaction to receive and process textual information. A user enters text commands into the control text file. See p. 10, lines 1-4 of paragraph [0027].), process the text-based commands written to the control file (Figure 3, block 308: Process the control and status files associated with the transaction. See p. 11, lines 1-3 of paragraph [0028].) and operate on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands (Figure 3, block 310: Operate on the pseudo-filesystem within the transaction and according to the text-based commands in the control file. See p. 11, lines 7-11 in paragraph [0028].).

Claim 22

An apparatus that interfaces with a filesystem, comprising (Figure 6 is a block diagram of a system 600 for performing methods of the present invention: see p. 16, lines 1-2 of paragraph [0044].):

a processor (Figure 6, block 606: p. 16, line 5 of paragraph [0044].);

a memory (Figure 6, block 602 shows a memory: p. 16, line 3 of paragraph [0044].) having instructions capable of being executed on the processor that receive a text-based command in a command file for operating on a pseudo-filesystem corresponding to the filesystem within a transaction (Figure 4, block 402: Receive a command indication in a control text file. A user enters a command in a control text file: see p. 13, lines 3-5 of paragraph [0034].), determine whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem (Figure 4, block 406: Determine whether data dependencies prevent the command from being executed. See p. 14, lines 1-2 of paragraph [0036].), perform the text-based command to update the pseudo-filesystem (Figure 4, block 410: Perform the command to affect files associated with the pseudo-filesystem. See p. 15, lines 1-2 of paragraph [0039].), update the filesystem to include changes performed to the pseudo-filesystem according to the text-based command (Figure 5, block 510: Perform the commit for files in the pseudo-filesystem and corresponding file system. See p. 16, lines 1-5 in paragraph [0043].), and update a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem (Figure 5, block 512: Enter status in to status file to indicate result of commit. See p. 16, lines 7-9 of paragraph [0043].).

Claim 23

An apparatus for creating a filesystem with transaction based functionality, comprising (Figure 6 is a block diagram of a system 600 for performing methods of the present invention: see p. 16, lines 1-2 of paragraph [0044].):

means for receiving (example means is textual filesystem interface 618 in Figure 6) an indicator to initiate a transaction for files stored in one or more portions of the filesystem (Figure 3, block 302: Receive an indicator to make directories under a pseudo-filesystem and start a transaction. See p. 9, lines 3-7 of paragraph [0024].);

means for duplicating (example means is transaction file system manager 620 in Figure 6) the filesystem within a pseudo-filesystem (Figure 3, block 304: Duplicate the filesystem within the pseudo-filesystem. A copy of the filesystem is created and mounted under the pseudo-file system. See p. 9, lines 1-3 of paragraph [0025].);

means for creating (example means is transaction file system manager 620 in Figure 6) a control file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem (Figure 3, block 306: Create control and status text files associated with the transaction to receive and process textual information. A user enters text commands into the control text file. See p. 10, lines 1-4 of paragraph [0027].);

means for processing (example means is processor 606 in Figure 6) the text-based commands written to the control file (Figure 3, block 308: Process the control and status files associated with the transaction. See lines p. 11, 1-3 of paragraph [0028].); and

means for operating (example means is pseudo-filesystem manager 624 in Figure 6) on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands (Figure 3, block 310: Operate on the pseudo-filesystem within the transaction and according to the text-based commands in the control file. See p. 11, lines 7-11 in paragraph [0028].).

Claim 24

An apparatus for interfacing with a filesystem, comprising (Figure 6 is a block diagram of a system 600 for performing methods of the present invention: see p. 16, lines 1-2 of paragraph [0044].):

means for receiving (example means is textual filesystem interface 618 in Figure 6) a text-based command in a command file for operating on a pseudo-filesystem corresponding to the filesystem within a transaction (Figure 4, block 402: Receive a command indication in a control text file. A user enters a command in a control text file: see p. 13, lines 3-5 of paragraph [0034].);

means for determining (example means is transaction file system manager 620 in Figure 6) whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem (Figure 4, block 406: Determine whether data dependencies prevent the command from being executed. See p. 14, lines 1-2 of paragraph [0036].);

means for performing (example means is processor 606 in Figure 6) the text-based command (Figure 4, block 410: Perform the command to affect files associated with the pseudo-filesystem. See p. 15, lines 1-2 of paragraph [0039].);

means for updating (example means is transaction file system manager 620 in Figure 6) the filesystem to include modifications performed to files and directories in the pseudo-filesystem (Figure 5, block 510: Perform the commit for files in the pseudo-filesystem and corresponding file system. See p. 16, lines 1-5 in paragraph [0043].); and

means for updating (example means is transaction file system manager 620 in Figure 6) a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem (Figure 5, block 512: Enter status in to status file to indicate result of commit. See p. 16, lines 7-9 of paragraph [0043].).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-2, 4-15, and 17-25 are rejected under 35 USC § 103(a) as being unpatentable over USPN 6,856,003 (Verma) in view of "CVS II: Parallelizing Software Development" (Berliner) and further in view of USPN 6,047,294 (Deshayes).

Claim 16 is rejected under 35 USC § 103(a) as being unpatentable over Verma in view of "On optimistic methods for concurrency control" (Kung).

VII. ARGUMENT

The rejection of claims 1 – 2 and 4 – 24 is improper, and Applicants respectfully request reversal of these rejections.

The claims do not stand or fall together. Instead, Applicants present separate arguments for various claims. Each of these arguments is separately argued below and presented with separate headings and three sub-heading as required by 37 C.F.R. § 41.37(c)(1)(vii).

Overview of Claims and Primary References (Verma, Berliner, & Deshayes)

As a precursor to the arguments, Applicants provide an overview of the claims and the primary references (Verma, Berliner, and Deshayes). This overview will assist in determining the scope and content of the prior art as required in Graham (see *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1, 17-18 setting out an objective analysis for applying 103 rejections).

As discussed in Applicants' Background, ensuring data consistency in a filesystem is a difficult problem when multiple users or processes access the same set of files. File sharing is even more difficult and complicated when using remote file sharing. Exemplary embodiments of the present invention solve these problems by creating and using a pseudo-filesystem. Changes to the files are performed in the pseudo-filesystem and then applied to the existing filesystem.

Specifically, the claims are directed to creating a filesystem that uses a textual interface (i.e., a user enters text commands into a control text file for the commands to be performed: see paragraph [0015] for example benefits of using text based commands.). A copy of the filesystem is created and mounted under a pseudo-file system (i.e., the filesystem is duplicated to a pseudo-filesystem). Thereafter, the text based commands are operated on the pseudo-file system. Once the transaction is complete, the original filesystem is updated with the changes made to the pseudo-filesystem.

Verma teaches a transactional file system wherein multiple file system operations are performed as a single transaction. A user can selectively control the scope and duration of a transaction within the file system. Verma does not teach or suggest

duplicating a filesystem into a pseudo-file system and using text based commands to operate on the pseudo-file system.

Berliner teaches a program (named "cvs") that enables multiple software developers to modify different modules of code for a software program without having such modifications cause conflicts. In other words, cvs enables "concurrent editing of source files among multiple software developers" (see section 2, paragraph 1, lines 5-6). Berliner does not teach or suggest duplicating a filesystem into a pseudo-file system and using text based commands to operate on the pseudo-file system.

Deshayes teaches methods and systems for backing up and restoring data in a computer system. Specifically, data from a primary storage device is copied to a backup storage device. When the data is being recovered, the data is restored to a scratch memory area. Deshayes does not teach or suggest duplicating a filesystem into a pseudo-file system and using text based commands to operate on the pseudo-file system. In other words, although Deshayes does backup data, Deshayes never teaches or even suggests this data is a duplicated filesystem.

Claim Rejections: 35 USC § 103(a)

Claims 1-2, 4-15, and 17-25 are rejected under 35 USC § 103(a) as being unpatentable over USPN 6,856,003 (Verma) in view of "CVS II: Parallelizing Software Development" (Berliner) and further in view of USPN 6,047,294 (Deshayes). These rejections are traversed.

Each of the independent claims recites one or more elements not taught or suggested in Verma in view of Berliner and Deshayes. These missing elements show that the differences between the combined teachings in the art and the recitations in the claims are great. As such, the pending claims are not a predictable variation of the art to one of ordinary skill in the art. Some examples are provided below for three different claim groups having separate sub-headings.

Sub-Heading: Group I

Group I includes independent claims 1, 19, 21, and 23 and their respective dependent claims. Claim 1 is selected for discussion.

As one example, claim 1 recites duplicating a filesystem within a pseudo-filesystem. The Examiner admits that Verma does not teach this element (see Final OA at p. 4). Applicants agree with this admission. The Examiner, however, attempts to cure this deficiency with Deshayes. Applicants respectfully disagree.

Deshayes teaches methods and systems for backing up and restoring data in a computer system. Specifically, data from a primary storage device is copied to a backup storage device. When the data is being recovered, the data is restored to a scratch memory area. Deshayes does not teach or suggest duplicating a filesystem into a pseudo-file system. In other words, although Deshayes does backup data, Deshayes never teaches or even suggests this data is a duplicated filesystem. Deshayes does not even mention duplicate filesystems or using a pseudo-filesystem as recited in claim 1.

Applicants acknowledge that claims must be given their broadest interpretation during patent examination. However, this interpretation must be a "**reasonable interpretation consistent with the specification**" (see MPEP 2111: emphasis added). Applicants' specification repeatedly uses the term "filesystem" in a manner consistent with the plain meaning of this term. It is not reasonable to interpret general usage of the word "data" in Deshayes as a "filesystem" or a "pseudo-filesystem." Deshayes is not concerned with duplicating filesystems within a pseudo-filesystem.

The differences between the claims and the teachings in the art are great since the references fail to teach or suggest all of the claim elements. As such, the pending claims are not a predictable variation of the art to one of ordinary skill in the art.

For at least these reasons, claims 1, 19, 21, and 23 and their respective dependent claims are allowable over Verma in view of Berliner and Deshayes.

As another example, claim 1 recites creating a control text file that provides a textual filesystem interface and receives text-based commands to operate on a pseudo-filesystem. The Examiner admits that Verma does not teach this element (see Final OA at p. 3). Applicants agree with this admission. The Examiner, however, attempts to cure this deficiency with Berliner. Applicants respectfully disagree.

Berliner teaches a program (named "cvs") that enables multiple software developers to modify different modules of code for a software program without having such modifications cause conflicts. In other words, cvs enables "concurrent editing of

source files among multiple software developers" (see section 2, paragraph 1, lines 5-6). Nowhere does Berliner teach or even suggest duplicating a filesystem within a pseudo-filesystem. Berliner does discuss mapping locations of software code, but this teaching in Berliner has nothing whatsoever to do with duplicating a filesystem. Since Berliner does not teach or suggest a pseudo-filesystem, therefore Berliner cannot teach or suggest a control text file that "receives text-based commands to operate on the pseudo-filesystem" as recited in claim 1.

The Examiner argues section 2.2 of Berliner. This section of Berlin teaches the use of scripts. These scripts, however, are not used to "operate on a pseudo-filesystem" as expressly recited in claim 1.

The differences between the claims and the teachings in the art are great since the references fail to teach or suggest all of the claim elements. As such, the pending claims are not a predictable variation of the art to one of ordinary skill in the art.

For at least these reasons, claims 1, 19, 21, and 23 and their respective dependent claims are allowable over Verma in view of Berliner and Deshayes.

As yet another example, claim 1 recites operating on one or more portions of the pseudo-file system within a transaction according to text-based commands. The Examiner argues that Verma teaches this element at column 3, lines 3-6. Applicants strongly disagree.

The Examiner previously admitted that Verma "does not teach duplicating the filesystem within a pseudo-filesystem" (see Final OA at p. 4). Since Verma does not teach the pseudo-filesystem, Verma therefore cannot teach the claim element "operating on one or more portions of the pseudo-filesystem." In other words, the Examiner admitted that Verma does not teach duplicating the filesystem within a pseudo-filesystem. A fortiori, Verma cannot then teach or suggest operating on the pseudo-filesystem.

Regardless, the Examiner cites column 3, lines 3-6 in Verma for allegedly teaching the claim element. This section of Verma discusses namespace and file data isolation. This section of Verma has nothing whatsoever to do with "operating on one or more portions of the pseudo-filesystem."

The differences between the claims and the teachings in the art are great since the references fail to teach or suggest all of the claim elements. As such, the pending claims are not a predictable variation of the art to one of ordinary skill in the art.

For at least these reasons, claims 1, 19, 21, and 23 and their respective dependent claims are allowable over Verma in view of Berliner and Deshayes.

Sub-Heading: Group II

Group II includes independent claims 12, 20, 22, and 24 and their respective dependent claims. Claim 12 is selected for discussion.

As one example, claim 12 recites performing a text-based command to modify a file in a pseudo-file system and then updating a filesystem to include modifications performed to the file in the pseudo-filesystem. Verma in view of Berliner and Deshayes do not teach or suggest this element.

Verma is directed to a transactional file system wherein multiple file system operations are performed as a transaction. Nowhere does Verma teach or even suggest two filesystems, namely a first filesystem and then a pseudo-filesystem wherein a file modified in the pseudo-file system is updated to the filesystem. Again, Verma teaches a single filesystem wherein plural operations are performed as part of a single user-level transaction.

Berliner teaches a program (named "cvs") that enables multiple software developers to modify different modules of code for a software program without having such modifications cause conflicts. In other words, cvs enables "concurrent editing of source files among multiple software developers" (see section 2, paragraph 1, lines 5-6). Nowhere does Berliner teach or even suggest two filesystems, namely a first filesystem and then a pseudo-filesystem wherein a file modified in the pseudo-file system is updated to the filesystem. Again, Berliner is directed modifying modules in a computer program.

Deshayes teaches methods and systems for backing up and restoring data in a computer system. Specifically, data from a primary storage device is copied to a backup storage device. When the data is being recovered, the data is restored to a scratch memory area. Nowhere does Deshayes teach or even suggest two filesystems, namely a first filesystem and then a pseudo-filesystem wherein a file modified in the pseudo-file

system is updated to the filesystem. Again, Deshayes is directed backing up data, but never suggests two filesystems.

The differences between the claims and the teachings in the art are great since the references fail to teach or suggest all of the claim elements. As such, the pending claims are not a predictable variation of the art to one of ordinary skill in the art.

For at least these reasons, claims 12, 20, 22, and 24 and their respective dependent claims are allowable over Verma in view of Berliner and Deshayes.

As yet another example, claim 12 recites updates a status file with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem. The Examiner has ignored this element. Instead of specifically addressing this claim element, the Examiner states "For the remaining steps of this claim applicant(s) is/are directed to the remarks and discussions made in claim 1 above" (see Final OA at p. 7). Claim 1, however, does not recite this element. Thus, the Examiner has failed to make a prima facie case of obviousness.

Further, applicants have reviewed Verma in view of Berliner and Deshayes but can find not teaching or suggest whatsoever for updating a status file with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem.

The differences between the claims and the teachings in the art are great since the references fail to teach or suggest all of the claim elements. As such, the pending claims are not a predictable variation of the art to one of ordinary skill in the art.

For at least these reasons, claims 12, 20, 22, and 24 and their respective dependent claims are allowable over Verma in view of Berliner and Deshayes.

Sub-Heading: Group III

Group III includes dependent claim 13.

Dependent claim 13 recites creating an entire copy of the filesystem and then mounting the entire copy of the filesystem under the pseudo-filesystem. The Examiner argues that these elements are taught in the abstract of Deshayes. Applicants respectfully disagree.

Deshayes teaches methods and systems for backing up and restoring data in a computer system. Specifically, data from a primary storage device is copied to a backup storage device. When the data is being recovered, the data is restored to a scratch memory area. Nowhere does Deshayes teach or even the data being backed up is a filesystem. Further, nowhere does Deshayes teach or suggest "mounting the entire copy of the filesystem under the pseudo-filesystem." Again, Deshayes generally teaches backing up data, not mounting filesystems.

The differences between claim 13 and the teachings in the art are great since the references fail to teach or suggest all of the claim elements. As such, claim 13is not a predictable variation of the art to one of ordinary skill in the art.

For at least these reasons, claim 13 is allowable over Verma in view of Berliner and Deshayes.

Claim Rejections: 35 USC § 103(a)

Claim 16 is rejected under 35 USC § 103(a) as being unpatentable over Verma in view of "On optimistic methods for concurrency control" (Kung). These rejections are traversed.

As shown above, Verma does not teach all of the elements of independent claim 12. Kung fails to cure these deficiencies. Thus for at least the reasons provided in connection with independent claim 12, dependent claim 16 is allowable over Verma in view of Kung.

CONCLUSION

In view of the above, Applicants respectfully request the Board of Appeals to reverse the Examiner's rejection of all pending claims.

Any inquiry regarding this Amendment and Response should be directed to Philip S. Lyren at Telephone No. 832-236-5529. In addition, all correspondence should continue to be directed to the following address:

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Respectfully submitted,

/Philip S. Lyren #40.709/

Philip S. Lyren
Reg. No. 40.709
Ph: 832-236-5529

VIII. Claims Appendix

1. A method of creating a filesystem with transaction based functionality, comprising:

receiving an indicator to initiate a transaction for files stored in one or more portions of the filesystem;

duplicating the filesystem within a pseudo-filesystem;

creating a control text file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem;

processing the text-based commands written to the control text file; and

operating on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands.

2. The method of claim 1 wherein the duplicating is performed lazily in order to reduce processing impact on the filesystem.

3. (canceled)

4. The method of claim 1 further comprising:

completing the transaction upon receipt of a text-based command associated with terminating the transaction.

5. The method of claim 3 wherein the text-based commands include functional equivalent commands associated with terminating the transaction and selected from a set of commands for performing one of the following functions: delete directory, delete

filesystem, and abort.

6. The method of claim 1 further comprising: updating the filesystem with updates performed on the pseudo-filesystem when the transaction has completed.

7. The method of claim 6 wherein the updates are performed upon receipt of an indication to commit the transaction.

8. The method of claim 1 further comprising:

creating a status text file that provides text-based status results from operations performed on the pseudo-filesystem.

9. The method of claim 1 wherein the indicator to initiate the transaction results from creation of a directory within the pseudo-filesystem.

10. The method of claim 1 wherein the transaction ensures atomic updates to the filesystem in accordance with modifications made to the pseudo-filesystem and related files during the transaction.

11. The method of claim 1 wherein a user assists in reconciliation of conflicts between updates in the pseudo-filesystems.

12. A method of interfacing with a filesystem comprising:

receiving a text-based command in a command file for operating on a pseudo-filesystem corresponding to the filesystem within a transaction;

determining whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem;

performing the text-based command to modify a file in the pseudo-filesystem;

updating the filesystem to include modifications performed to the file in the pseudo-filesystem; and

updating a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem.

13. The method of claim 12 further comprising:

creating an entire copy of the filesystem;

mounting the entire copy of the filesystem under the pseudo-filesystem.

14. The method of claim 12 further comprising:

creating a textual interface;

receiving the text-based command from a user into the textual interface.

15. The method of claim 12 wherein receiving a text-based command includes functional equivalent commands selected from a set including: change root directory, select concurrency control type, select isolation level, commit transaction, and abort transaction.

16. The method of claim 12 wherein determining the one or more data dependencies includes using optimistic concurrency control (OCC) to control pending read and write operations to the pseudo-filesystem, the filesystem and one or more corresponding files associated with the pseudo-filesystem and filesystem respectively.

17. The method of claim 12 wherein determining the one or more data dependencies includes using lock-based concurrency control (LBCC) to control pending read and write operations to the pseudo-filesystem, the filesystem and one or more corresponding files associated with the pseudo-filesystem and filesystem respectively.

18. The method of claim 12 wherein a user assists in reconciliation of conflicts between resources in the filesystem and pseudo-filesystems and files associated with these.

19. A computer program product for creating a filesystem with transaction based functionality, tangibly stored on a computer readable medium, comprising instructions operable to cause a programmable processor to:

receive an indicator to initiate a transaction for files stored in one or more portions of the filesystem;

duplicate the filesystem within a pseudo-filesystem;

create a control file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem;

process the text-based commands written to the control file; and

operate on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands.

20. A computer program product for interfacing with a filesystem, tangibly stored on a computer readable medium, comprising instructions operable to cause a programmable processor to:

receive a text-based command in a command file for operating on a pseudo-filesystem that is a copy of the filesystem;

determine whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem;

perform the text-based command to update the pseudo-filesystem;

update the filesystem to include modifications made to the pseudo-filesystem according to the text-based command; and

update a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem.

21. An apparatus that creates a filesystem with transaction based functionality comprising:

a processor;

a memory having instructions capable of being executed on the processor that receive an indicator to initiate a transaction for files stored in one or more portions of the filesystem, duplicate the filesystem within a pseudo-filesystem create a control file that provides a textual filesystem interface for receiving text-based commands to operate on

the pseudo-filesystem, process the text-based commands written to the control file and operate on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands.

22. An apparatus that interfaces with a filesystem, comprising:

a processor;

a memory having instructions capable of being executed on the processor that receive a text-based command in a command file for operating on a pseudo-filesystem corresponding to the filesystem within a transaction, determine whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem, perform the text-based command to update the pseudo-filesystem, update the filesystem to include changes performed to the pseudo-filesystem according to the text-based command, and update a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem.

23. An apparatus for creating a filesystem with transaction based functionality, comprising:

means for receiving an indicator to initiate a transaction for files stored in one or more portions of the filesystem;

means for duplicating the filesystem within a pseudo-filesystem;

means for creating a control file that provides a textual filesystem interface and receives text-based commands to operate on the pseudo-filesystem;

means for processing the text-based commands written to the control file; and

means for operating on one or more portions of the pseudo-filesystem within a transaction according to the text-based commands.

24. An apparatus for interfacing with a filesystem, comprising:

means for receiving a text-based command in a command file for operating on a pseudo-filesystem corresponding to the filesystem within a transaction;

means for determining whether one or more data dependencies would prevent the text-based command from being performed on the pseudo-filesystem;

means for performing the text-based command;

means for updating the filesystem to include modifications performed to files and directories in the pseudo-filesystem; and

means for updating a status file associated with the pseudo-filesystem with a text-based status result for performing the text-based command and updates performed in the filesystem.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.